

[illegible]

```

LL               IIIIII             SSSSSSSS
LL               IIIIII             SSSSSSSS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SSSSSS
LL               II                  SSSSSS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS

```



```
1 0001 0 MODULE setproces ( IDENT = 'V04-000',
2 0002 0 ADDRESSING_MODE (EXTERNAL = GENERAL)) =
3 0003 1 BEGIN
4 0004 1
5 0005 1 *****
6 0006 1 *
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
9 0009 1 * ALL RIGHTS RESERVED.
10 0010 1 *
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
16 0016 1 * TRANSFERRED.
17 0017 1 *
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
20 0020 1 * CORPORATION.
21 0021 1 *
22 0022 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
23 0023 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
24 0024 1 *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1 ++
29 0029 1 FACILITY: SETPRO Command
30 0030 1
31 0031 1 ABSTRACT:
32 0032 1
33 0033 1 This module sets various parameters for a process.
34 0034 1
35 0035 1 ENVIRONMENT:
36 0036 1
37 0037 1 VAX/VMS operating system, user mode
38 0038 1
39 0039 1 AUTHOR: Gerry Smith 12-Jan-1983
40 0040 1
41 0041 1 Modified by:
42 0042 1
43 0043 1 V03-007 AEW0002 Anne Warner 05-Jul-1984
44 0044 1 Change ALTPRV to ALTPRI from previous fix because
45 0045 1 there's no such creature as ALPTRV.
46 0046 1
47 0047 1 V03-006 AEW0001 Anne Warner 04-Jun-1984
48 0048 1 Add non-fatal error message SET$NOPRIO indicating
49 0049 1 that the process priority could not be raised above
50 0050 1 base priority because it does not have the user
51 0051 1 privilege ALTPRV. (actually its ALTPRI - see above)
52 0052 1
53 0053 1 V03-005 GAS0182 Gerry Smith 19-Sep-1983
54 0054 1 Change the way that privileges get set. Instead of
55 0055 1 disabling all privileges and then re-enabling them,
56 0056 1 figure out which privileges to enable, and which
57 0057 1 to disable, and then do it explicitly.
```


SETPROCES
V04-000

F 3
16-Sep-1984 00:45:54
14-Sep-1984 12:09:16

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETPROCES.B32;1

Page 2
(1)

: 58 0058 1 :
: 59 0059 1 :
: 60 0060 1 :
: 61 0061 1 :
: 62 0062 1 :
: 63 0063 1 :
: 64 0064 1 :
: 65 0065 1 :
: 66 0066 1 :
: 67 0067 1 :
: 68 0068 1 :
: 69 0069 1 :
: 70 0070 1 :
: 71 0071 1 :
: 72 0072 1 :
: 73 0073 1 :--

V03-004 GAS0157 Gerry Smith 25-Jul-1983
Use the real process ID of the process, rather than
what the user input.

V03-003 WMC0001 Wayne Cardoza 11-Apr-1983
Add SET PROC/DUMP.

V03-002 GAS0113 Gerry Smith 30-Mar-1983
Collect and validate all qualifiers and values first,
then make all the modifications.

V03-001 GAS0112 Gerry Smith 29-Mar-1983
Remove last traces of the old command dispatcher.

SETPROCES
V04-000

G 3
16-Sep-1984 00:45:54
14-Sep-1984 12:09:16

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETPROCES.B32;1

Page 3
(2)

```
: 75      0074 1 !  
: 76      0075 1 ! Include files  
: 77      0076 1 !  
: 78      0077 1 LIBRARY 'SYS$LIBRARY:LIB';  
: 79      0078 1  
: 80      0079 1
```

! VAX/VMS common definitions


```

82 0080 1  |
83 0081 1  | Table of contents
84 0082 1  |
85 0083 1  |
86 0084 1  FORWARD ROUTINE
87 0085 1      set$process : NOVALUE,      | Main routine
88 0086 1      get_name : NOVALUE,         | Get process name
89 0087 1      get_qual : NOVALUE,         | Get all qualifiers and values
90 0088 1      set_process : NOVALUE,      | Set them
91 0089 1      set_dump  : NOVALUE;       | Kernel mode routine to set dump flag
92 0090 1  |
93 0091 1  |
94 0092 1  | External routines
95 0093 1  |
96 0094 1  EXTERNAL ROUTINE
97 0095 1      lib$cvb_hb,                 | Convert ASCII (hex) to binary
98 0096 1      lib$cvb_db,                 | Convert ASCII (decimal) to binary
99 0097 1      prv$setpriv,                | Set/clear privilege bits in bitmask
100 0098 1      cli$get_value,             | Get value from CLI
101 0099 1      cli$present;               | See if qualifier is present
102 0100 1  |
103 0101 1  |
104 0102 1  | External globals
105 0103 1  |
106 0104 1  EXTERNAL
107 0105 1      ctl$gq_procpriv : VECTOR[2], | Process privileges
108 0106 1      ctl$gl_phd  : REF BLOCK[,BYTE]; | P1 window to PHD
109 0107 1  |
110 0108 1  |
111 0109 1  | Declare the final status return.
112 0110 1  |
113 0111 1  EXTERNAL
114 0112 1      set$exit_status;
115 0113 1  |
116 0114 1  |
117 0115 1  |
118 0116 1  | Declare some shared messages
119 0117 1  |
120 P 0118 1  $SHR_MSGDEF      (SET,119,LOCAL,
121 0119 1      (invquaval, error));
122 0120 1  |
123 0121 1  |
124 0122 1  | Declare literals defined elsewhere
125 0123 1  |
126 0124 1  EXTERNAL LITERAL
127 0125 1      cli$_absent,               | Qualifier absent
128 0126 1      cli$_negated,              | Qualifier explicitly negated
129 0127 1      set$_writeerr,             | Error modifying
130 0128 1      set$_noprio,                | Priority not changed
131 0129 1      set$_prio,                  | Priority changed
132 0130 1      set$_noname,                | Name not changed
133 0131 1      set$_nameset,               | Name changed
134 0132 1      set$_notsuspnd,             | Process not suspended
135 0133 1      set$_resumed,               | Process resumed
136 0134 1      set$_notresumed,            | Process not resumed
137 0135 1      set$_suspnd,                | Process suspended
138 0136 1      set$_modeset,               | Process mode changed
```



```

: 139      0137 1      set$_notpriv,      ! Privileges not set
: 140      0138 1      set$_privset,      ! Privileges set
: 141      0139 1      set$_ownproc;      ! Qualifier only good for own process
: 142      0140 1
: 143      0141 1
: 144      0142 1      ! Declare the literals for the different qualifiers
: 145      0143 1
: 146      0144 1      LITERAL
: 147      P 0145 1      $EQUList (set$_.,1,1,
: 148      P 0146 1          (log.),
: 149      P 0147 1          (priority.),
: 150      P 0148 1          (name.),
: 151      P 0149 1          (resume.),
: 152      P 0150 1          (suspend.),
: 153      P 0151 1          (swap.),
: 154      P 0152 1          (swapval.),
: 155      P 0153 1          (wait.),
: 156      P 0154 1          (waitval),
: 157      P 0155 1          (priv),
: 158      P 0156 1          (dump),
: 159      0157 1          (dumpval));
: 160      0158 1
: 161      0159 1
: 162      0160 1      ! It is convenient to declare one large vector containing all the data,
: 163      0161 1      ! and give the separate pieces names that humans like. So, declare a
: 164      0162 1      ! macro that will make those binds at the beginning of each subroutine.
: 165      0163 1
: 166      0164 1      MACRO
: 167      M 0165 1      BIND DATA =
: 168      M 0166 1      BIND
: 169      M 0167 1          flags      = data_buffer[0] : BITVECTOR[32],
: 170      M 0168 1          pid        = data_buffer[1] : VOLATILE,
: 171      M 0169 1          priority   = data_buffer[2],
: 172      M 0170 1          new_name    = data_buffer[3] : VECTOR[2],
: 173      M 0171 1          enab_priv  = data_buffer[5] : VECTOR[2],
: 174      M 0172 1          disab_priv = data_buffer[7] : VECTOR[2],
: 175      M 0173 1          name_desc  = data_buffer[9] : VECTOR[2],
: 176      0174 1          name_buffer = data_buffer[11] : VECTOR[3];%

```



```
178 0175 1 GLOBAL ROUTINE set$process : NOVALUE =
179 0176 2 BEGIN
180 0177 2 ++
181 0178 2 Functional description
182 0179 2
183 0180 2 This is the routine for the SET PROCESS command. It is called from the
184 0181 2 SET command processor, and sets various runtime parameters for a
185 0182 2 process.
186 0183 2
187 0184 2 Inputs
188 0185 2 None
189 0186 2
190 0187 2 Outputs
191 0188 2 None
192 0189 2
193 0190 2 ----
194 0191 2
195 0192 2 LOCAL
196 0193 2 status, ! Status return
197 0194 2 data_buffer : VECTOR[20] ! Buffer containing all the data
198 0195 2 INITIAL(REP 20 of (0)); ! initially clear
199 0196 2
200 0197 2 get_name(data_buffer); ! Get the name of the process.
201 0198 2
202 0199 2 get_qualifiers(data_buffer); ! Get all the qualifiers.
203 0200 2
204 0201 2 IF .set$exit_status ! If no errors so far,
205 0202 2 THEN set_process(data_buffer); ! set the new values.
206 0203 2
207 0204 2 RETURN;
208 0205 1 END;
```

```
.TITLE SETPROCES
.IDENT \V04-000\

.PSECT $SPLITS,NOWRT,NOEXE,2

00000000# 00000 P.AAA: .LONG 0[20]

.EXTRN LIB$CVT_HTB, LIB$CVT_DTB
.EXTRN PRV$SETPRIV, CLIS$GET-VALUE
.EXTRN CLIS$PRESENT, CTL$GQ_PROCPRIV
.EXTRN CTL$GQ_PHD, SET$EXIT STATUS
.EXTRN CLIS$ABSENT, CLIS$NEGATED
.EXTRN SET$WRITEERR, SET$NOPRIO
.EXTRN SET$PRIOSET, SET$NONAME
.EXTRN SET$NAMESET, SET$NOTSUSPND
.EXTRN SET$RESUMED, SET$NOTRESUMED
.EXTRN SET$SUSPND, SET$MODESET
.EXTRN SET$NOTPRIV, SET$PRIVSET
.EXTRN SET$OWNPROC

.PSECT $CODE$,NOWRT,2

5E B0 AE 003C 00000 .ENTRY SET$PROCESS, Save R2,R3,R4,R5 : 0175
          9E 00002 MOVAB -80(SP), SP :
```


SETPROCES
V04-000

K 3
16-Sep-1984 00:45:54
14-Sep-1984 12:09:16

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETPROCES.B32;1

Page 7
(4)

6E	0000'	CF	0050	8F	28	00006	MOV C3	#80, P.AAA, DATA_BUFFER	:	0195
				5E	DD	0000E	PUSHL	SP	:	0197
	0000V	CF		01	FB	00010	CALLS	#1, GET_NAME	:	
				5E	DD	00015	PUSHL	SP	:	0199
	0000V	CF		01	FB	00017	CALLS	#1, GET_QUALS	:	
		07	00000000G	00	E9	0001C	BLBC	SET\$EXIT_STATUS, 1\$:	0201
				5E	DD	00023	PUSHL	SP	:	0202
	0000V	CF		01	FB	00025	CALLS	#1, SET_PROCESS	:	
				04	0002A	1\$:	RET		:	0205

; Routine Size: 43 bytes, Routine Base: \$CODE\$ + 0000

```
210 0206 1 ROUTINE get_name (data_buffer) : NOVALUE =
211 0207 2 BEGIN
212 0208 2
213 0209 2 ++
214 0210 2
215 0211 2 Get the process name and tuck it away to use later.
216 0212 2
217 0213 2 Inputs
218 0214 2 DATA_BUFFER - contains all the data cells
219 0215 2
220 0216 2 Outputs
221 0217 2 NAME_DESC will point to the process name
222 0218 2 PID will contain the process ID of the process to change
223 0219 2
224 0220 2 --
225 0221 2
226 0222 2 MAP
227 0223 2 data_buffer : REF VECTOR;
228 0224 2
229 0225 2 LOCAL
230 0226 2 status, ; General status return
231 0227 2 desc : $BBLOCK[dsc$_s_bln], ; General descriptor
232 0228 2 iosb : VECTOR[4,WORD], ; Status block for GETJPI
233 0229 2 jpi_list : $ITMLST_DECL(ITEMS = 2); ; Item list for GETJPI
234 0230 2
235 0231 2
236 0232 2 Bind the data to names we can understand
237 0233 2
238 0234 2 bind_data;
239 0235 2
240 0236 2
241 0237 2 Collect the process name, if specified. If no process name is
242 0238 2 specified, try a process id.
243 0239 2
244 0240 2 $init_dyndesc(desc); ; Make the descriptor dynamic
245 0241 2 pid = 0; ; Show that no PID found yet.
246 0242 2 name_desc[1] = name_buffer; ; Point to process name buffer
247 0243 2
248 0244 2
249 0245 2 If the process name is given, also get the PID
250 0246 2
251 0247 2 IF cli$get_value(%ASCID 'PROCESS', desc) ! Get the process name
252 0248 2 THEN ; If the process name exists,
253 0249 2 BEGIN ; convert it to a PID.
254 0250 2
255 0251 2 Set up the JPI item list to get the PID.
256 0252 2
257 P 0253 2 $ITMLST_INIT(ITMLST = jpi_list,
258 0254 2 (ITMCOB = jpi$_pid, BUFADR = pid));
259 P 0255 2 status = $GETJPIW(ITMLST = jpi_list,
260 0256 2 PRCNAM = desc,
261 0257 2 IOSB = iosb);
262 0258 2
263 0259 2 IF .status
264 0260 2 THEN status = .iosb[0];
265 0261 2 IF NOT .status
266 0262 2 THEN SIGNAL(set$_writeerr, 1, desc, .status)
266 0262 2 ELSE
```



```
267      BEGIN
268      CH$MOVE(.desc[dsc$w_length], .desc[dsc$a_pointer], name_buffer);
269      name_desc[0] = .desc[dsc$w_length];
270      name_desc[1] = name_buffer;
271      END;
272      END
273      ---
274      If no process name, perhaps the PID was specified.
275      ELSE
276      BEGIN
277      IF cli$get_value(%ASCID 'IDENTIFICATION',
278      desc)
279      THEN
280      BEGIN
281      IF NOT (status = lib$cvtt_hb(.desc[dsc$w_length],
282      .desc[dsc$a_pointer],
283      pid))
284      THEN SIGNAL(set$_invquaval, 2,
285      desc, %ASCID 'IDENTIFICATION')
286      ELSE
287      BEGIN
288      $ITMLST_INIT(ITMLST = jpi_list,
289      (ITMCO = jpi$_pid,
290      BUFADR = pid),
291      (ITMCO = jpi$_prcnam,
292      BUFADR = name_buffer,
293      BUFSIZ = 20,
294      RETLEN = name_desc[0]));
295      status = $GETJPIW(ITMLST = jpi_list,
296      PIDADR = pid,
297      IOSB = iosb);
298      IF .status
299      THEN status = .iosb[0];
300      IF NOT .status
301      THEN SIGNAL(set$_writeerr, 1, desc, .status);
302      END;
303      END;
304      END;
305      ---
306      If no PID specified, use the PID and name of the current process.
307      IF .pid EQL 0
308      THEN
309      BEGIN
310      $ITMLST_INIT(ITMLST = jpi_list,
311      (ITMCO = jpi$_pid,
312      BUFADR = pid),
313      (ITMCO = jpi$_prcnam,
314      BUFADR = name_buffer,
315      BUFSIZ = 20,
316      RETLEN = name_desc[0]));
317      status = $GETJPIW(ITMLST = jpi_list,
318      IOSB = iosb);
319      IF .status
320      THEN status = .iosb[0];
321      END;
322      END;
323
```

```
! Set up JPI list to get
! the current process
! PID and name, and
! stuff them into the
! appropriate places.
```

```
: 324      0320 3      IF NOT .status
: 325      0321 3      THEN SIGNAL(set$_writeerr, 1, %ASCII 'this process', .status);
: 326      0322 2      END;
: 327      0323 2
: 328      0324 2      RETURN;
: 329      0325 1      END;
```

```
                                .PSECT $SPLITS,NOWRT,NOEXE,2
                                00 53 53 45 43 4F 52 50 00050 P.AAC: .ASCII \PROCESS\<0>
                                010E0007 00058 P.AAB: .LONG 17694727
                                00000000' 0005C P.AAE: .ADDRESS P.AAC
00 4E 4F 49 54 41 43 49 46 49 54 4E 45 44 49 00060 P.AAE: .ASCII \IDENTIFICATION\<0><0>
                                00 0006F
                                010E000E 00070 P.AAD: .LONG 17694734
                                00000000' 00074 P.AAE: .ADDRESS P.AAE
00 4E 4F 49 54 41 43 49 46 49 54 4E 45 44 49 00078 P.AAG: .ASCII \IDENTIFICATION\<0><0>
                                00 00087
                                010E000E 00088 P.AAF: .LONG 17694734
                                00000000' 0008C P.AAG: .ADDRESS P.AAG
                                73 73 65 63 6F 72 70 20 73 69 68 74 00090 P.AAI: .ASCII \this process\
                                010E000C 0009C P.AAH: .LONG 17694732
                                00000000' 000A0 P.AAH: .ADDRESS P.AAI
```

.EXTRN SYSS\$GETJPIW

.PSECT \$CODE\$,NOWRT,2

```
OFFC 00000 GET_NAME:
5B 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 0206
5A 00000000G 00 9E 00009 MOVAB CLIS$GET_VALUE, R11
5E          2C C2 00010 MOVAB SYSS$GETJPIW, R10
50          04 AC D0 00013 SUBL2 #44, SP : 0229
58          04 A0 9E 00017 MOVAB 4(R0), R8
57          24 A0 9E 0001B MOVAB 36(R0), R7
59          2C A0 9E 0001F MOVAB 44(R0), R9
24 AE 020E0000 8F D0 00023 MOVAB #34471936, DESC : 0240
          28 AE D4 0002B CLRL DESC+4
          68 D4 0002E CLRL (R8) : 0241
04 A7          59 D0 00030 MOVAB R9, 4(R7) : 0242
          24 AE 9F 00034 PUSHAB DESC : 0247
          0000' CF 9F 00037 PUSHAB P.AAB
          6B          02 FB 0003B CALLS #2, CLIS$GET_VALUE
          3D          50 E9 0003E BLBC R0, 1$
          50          6E 9E 00041 MOVAB JPI_LIST, $$ITMBLKPTR : 0254
          80 03190004 8F D0 00044 MOVAB #51970052, ($$ITMBLKPTR)+
          80          58 D0 0004B MOVAB R8, ($$ITMBLKPTR)+
          80          7C 0004E CLRQ ($$ITMBLKPTR)+
          7E 7C 00050 CLRQ -(SP) : 0257
          24 AE 9F 00052 PUSHAB IOSB
          0C AE 9F 00055 PUSHAB JPI_LIST
          34 AE 9F 00058 PUSHAB DESC
          7E 7C 0005B CLRQ -(SP)
          6A          07 FB 0005D CALLS #7, SYSS$GETJPIW
```


Page 11
(5)

SETPROCES
V04-000

C 4
16-Sep-1984 00:45:54
14-Sep-1984 12:09:16

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETPROCES.B32;1

Page 12
(5)

		7E	7C	00124	CLRQ	-(SP)		
	24	AE	9F	00126	PUSHAB	IOSB		: 0317
	0C	AE	9F	00129	PUSHAB	JPI LIST		: :
		7E	7C	0012C	CLRQ	-(SP)		: :
		7E	D4	0012E	CLRL	-(SP)		: :
6A		07	FB	00130	CALLS	#7, SYSSGETJPIW		: :
56		50	D0	00133	MOVL	R0, STATUS		: :
07		56	E9	00136	BLBC	STATUS, 7\$: 0318
56	1C	AE	3C	00139	MOVZWL	IOSB, STATUS		: 0319
15		56	E8	0013D	BLBS	STATUS, 8\$: 0320
		56	DD	00140	PUSHL	STATUS		: 0321
	0000'	CF	9F	00142	PUSHAB	P.AAH		: :
		01	DD	00146	PUSHL	#1		: :
00000000G	00	8F	DD	00148	PUSHL	#SET\$ WRITEERR		: :
	00000000G	04	FB	0014E	CALLS	#4, LIB\$SIGNAL		: :
		04	00155	8\$:	RET			: 0325

; Routine Size: 342 bytes, Routine Base: \$CODE\$ + 002B


```
331 0326 1 ROUTINE get_qual (data_buffer) : NOVALUE =
332 0327 2 BEGIN
333 0328 2 ++
334 0329 2
335 0330 2 Get all and validate all the qualifiers. If any errors, signal them.
336 0331 2
337 0332 2 Inputs
338 0333 2 DATA_BUFFER contains all the data cells.
339 0334 2
340 0335 2 Outputs
341 0336 2 FLAGS will have bits set to indicate what is to change.
342 0337 2 PRIORITY will have the new priority.
343 0338 2 NEW NAME will point to the new process name.
344 0339 2 PRIV will be the new privilege mask.
345 0340 2
346 0341 2
347 0342 2
348 0343 2 Bind the data buffer to names that humans like.
349 0344 2
350 0345 2
351 0346 2 MAP
352 0347 2 data_buffer : REF VECTOR;
353 0348 2
354 0349 2 LOCAL
355 0350 2 status,
356 0351 2 ourpid,
357 0352 2 iosb : VECTOR[2],
358 0353 2 jpi_list : $ITMLST DECL(ITEMS = 1),
359 0354 2 desc : $BBLOCK[dsc$c_s_bln];
360 0355 2
361 0356 2
362 0357 2 Bind the data buffer to names that are more understandable
363 0358 2
364 0359 2 bind_data:
365 0360 2
366 0361 2
367 0362 2 Obtain the process ID of this process. It will be used to check that
368 0363 2 certain qualifiers are not requested inappropriately.
369 0364 2
370 P 0365 2 $ITMLST_INIT(ITMLST = jpi_list,
371 P 0366 2 (ITMCOD = jpi$pid,
372 P 0367 2 BUFADR = ourpid)
373 0368 2 );
374 P 0369 2 $GETJPIW(ITMLST = jpi_list,
375 0370 2 IOSB = iosb);
376 0371 2
377 0372 2
378 0373 2
379 0374 2 See if logging is requested.
380 0375 2
381 0376 2 flags[set$log] = cli$present(%ASCID 'LOG');
382 0377 2
383 0378 2
384 0379 2 /PRIORITY=n
385 0380 2
386 0381 2 $init_dyndesc(desc);
387 0382 2 IF cli$get_value(%ASCID 'PRIORITY', desc)
```

```
! Make desc. dynamic
! See if qualifier there
```



```
388 0383 2 THEN
389 0384 BEGIN
390 0385 flags[set$priority] = 1;
391 0386 IF NOT lib$cvtdtb(.desc[dsc$w_length],           ! If not a good value,
392 0387 .desc[dsc$a_pointer],           ! tell the user
393 0388 priority)
394 0389 THEN SIGNAL(set$_invquaval, 2, desc, %ASCII 'PRIORITY')
395 0390 ELSE
396 0391 BEGIN           ! Perform bounds
397 0392 IF .priority GTR 31           ! checking, telling
398 0393 OR .priority LSS 0           ! if out of bounds
399 0394 THEN SIGNAL(set$_invquaval, 2, desc, %ASCII 'PRIORITY');
400 0395 END;
401 0396 END;
402 0397
403 0398 !
404 0399 /NAME = string
405 0400
406 0401 IF cli$get_value(%ASCII 'NAME', desc)           ! If a new name requested
407 0402 THEN
408 0403 BEGIN
409 0404 IF .ourpid NEQ .pid
410 0405 THEN SIGNAL(set$_ownproc,
411 0406 1,
412 0407 %ASCII 'NAME');
413 0408 flags[set$name] = 1;           ! Set the flag
414 0409 new_name[0] = .desc[dsc$w_length];           ! Point to the name
415 0410 new_name[1] = .desc[dsc$a_pointer];
416 0411 $init_dyndesc(desc);           ! Re-use the descriptor
417 0412 END;
418 0413
419 0414 !
420 0415 /SUSPEND and /RESUME are inverses of each other, and so are treated
421 0416 together. However, although there is a /NOSUSPEND, there is no /NORESUME.
422 0417
423 0418 status = cli$present(%ASCII 'SUSPEND');
424 0419 IF .status
425 0420 THEN flags[set$_suspend] = 1
426 0421 ELSE IF .status EQL cli$_negated
427 0422 THEN flags[set$_resume] = 1;
428 0423 IF cli$present(%ASCII 'RESUME')
429 0424 THEN flags[set$_resume] = 1;
430 0425
431 0426 !
432 0427 /[NO]SWAP
433 0428
434 0429 status = cli$present(%ASCII 'SWAPPING');
435 0430 IF .status NEQ cli$_absent
436 0431 THEN
437 0432 BEGIN
438 0433 IF .ourpid NEQ .pid
439 0434 THEN SIGNAL(set$_ownproc,
440 0435 1,
441 0436 %ASCII '[NO]SWAP');
442 0437 flags[set$_swap] = 1;
443 0438 flags[set$_swapval] = NOT .status;
444 0439 END;
```



```

445 0440 2
446 0441 2
447 0442 2 / [NO]RESOURCE_WAIT
448 0443 2
449 0444 2 status = cli$present(%ASCID 'RESOURCE_WAIT');
450 0445 2 IF .status NEQ cli$_absent
451 0446 2 THEN
452 0447 2 BEGIN
453 0448 2 IF .ourpid NEQ .pid
454 0449 2 THEN SIGNAL(set$_ownproc,
455 0450 2 1,
456 0451 2 %ASCID '[NO]RESOURCE_WAIT');
457 0452 2 flags[set$_wait] = 1;
458 0453 2 flags[set$_waitval] = NOT .status;
459 0454 2 END;
460 0455 2
461 0456 2 /PRIVILEGES = list
462 0457 2
463 0458 2 IF cli$present(%ASCID 'PRIVILEGES')
464 0459 2 THEN
465 0460 2 BEGIN
466 0461 2 LOCAL
467 0462 2 oldpriv : VECTOR[2],
468 0463 2 newpriv : VECTOR[2];
469 0464 2 IF .ourpid NEQ .pid
470 0465 2 THEN SIGNAL(set$_ownproc,
471 0466 2 1,
472 0467 2 %ASCID 'PRIVILEGES');
473 0468 2 flags[set$_priv] = 1;
474 0469 2
475 0470 2
476 0471 2
477 0472 2 Copy the current process privileges into local memory.
478 0473 2
479 0474 2 oldpriv[0] = newpriv[0] = .ctl$gq_procpriv[0];
480 0475 2 oldpriv[1] = newpriv[1] = .ctl$gq_procpriv[1];
481 0476 2
482 0477 2
483 0478 2 Then get all the privileges that were specified by the user. For
484 0479 2 each privilege given, call the unsupported, undocumented routine
485 0480 2 PRV$SETPRIV, which will decipher the ASCII text given it (e.g. NOLOG)
486 0481 2 and set or clear the corresponding bit in the two-longword privilege
487 0482 2 bitmask.
488 0483 2
489 0484 2 WHILE cli$get_value(%ASCID 'PRIVILEGES', desc)
490 0485 2 DO
491 0486 2 BEGIN
492 0487 2 IF NOT (status = PRV$SETPRIV(desc, newpriv))
493 0488 2 THEN SIGNAL(set$_invquaval, 2, ! Say it's invalid
494 0489 2 desc,
495 0490 2 %ASCID 'PRIVILEGES');
496 0491 2 END;
497 0492 2
498 0493 2
499 0494 2 Get the privileges to enable and disable.
500 0495 2
501 0496 2 enab_priv[0] = .newpriv[0] AND NOT .oldpriv[0];
```

```

: 502      0497      3      enab_priv[1] = .newpriv[1] AND NOT .oldpriv[1];
: 503      0498      3      disab_priv[0] = .oldpriv[0] AND NOT .newpriv[0];
: 504      0499      3      disab_priv[1] = .oldpriv[1] AND NOT .newpriv[1];
: 505      0500      3      END;
: 506      0501      3
: 507      0502      3      !
: 508      0503      3      /[[NO]DUMP
: 509      0504      3      !
: 510      0505      3      status = cli$present(%ASCID 'DUMP');
: 511      0506      3      IF .status NEQ cli$_absent
: 512      0507      3      THEN
: 513      0508      3      BEGIN
: 514      0509      3      IF .ourpid NEQ .pid
: 515      0510      3      THEN SIGNAL(set$_ownproc,
: 516      0511      3      1,
: 517      0512      3      %ASCID 'DUMP');
: 518      0513      3      flags[set$_dump] = 1;
: 519      0514      3      flags[set$_dumpval] = .status;
: 520      0515      3      END;
: 521      0516      3
: 522      0517      2      RETURN;
: 523      0518      1      END;

```

INFO#250 LI:0404
Referenced LOCAL symbol OURPID is probably not initialized

```

.PSECT $PLITS,NOWRT,NOEXE,2
00 47 4F 4C 000A4 P.AAK: .ASCII \LOG\<0>
010E0003 000A8 P.AAJ: .LONG 17694723
00000000' 000AC P.AAM: .ADDRESS P.AAK
59 54 49 52 4F 49 52 50 000B0 P.AAM: .ASCII \PRIORITY\
010E0008 000B8 P.AAL: .LONG 17694728
00000000' 000BC P.AAM: .ADDRESS P.AAM
59 54 49 52 4F 49 52 50 000C0 P.AAO: .ASCII \PRIORITY\
010E0008 000C8 P.AAN: .LONG 17694728
00000000' 000CC P.AAO: .ADDRESS P.AAO
59 54 49 52 4F 49 52 50 000D0 P.AAQ: .ASCII \PRIORITY\
010E0008 000D8 P.AAP: .LONG 17694728
00000000' 000DC P.AAQ: .ADDRESS P.AAQ
45 4D 41 4E 000E0 P.AAS: .ASCII \NAME\
010E0004 000E4 P.AAR: .LONG 17694724
00000000' 000E8 P.AAS: .ADDRESS P.AAS
45 4D 41 4E 000EC P.AAU: .ASCII \NAME\
010E0004 000F0 P.AAT: .LONG 17694724
00000000' 000F4 P.AAU: .ADDRESS P.AAU
00 44 4E 45 50 53 55 53 000F8 P.AAW: .ASCII \SUSPEND\<0>
010E0007 00100 P.AAV: .LONG 17694727
00000000' 00104 P.AAW: .ADDRESS P.AAW
00 00 45 4D 55 53 45 52 00108 P.AAY: .ASCII \RESUME\<0><0>
010E0006 00110 P.AAX: .LONG 17694726
00000000' 00114 P.AAY: .ADDRESS P.AAY
47 4E 49 50 50 41 57 53 00118 P.ABA: .ASCII \SWAPPING\
010E0008 00120 P.AAZ: .LONG 17694728
00000000' 00124 P.ABA: .ADDRESS P.ABA
50 41 57 53 5D 4F 4E 5B 00128 P.ABC: .ASCII \[[NO]SWAP\

```


SETPROCES
V04-000

H 4
16-Sep-1984 00:45:54 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:16 [CLIUTL.SRC]SETPROCES.B32;1

Page 17
(6)

```
00 00 54 49 41 57 5F 45 43 52 55 4F 53 45 52 00130 P.ABB: .LONG 17694728
00000000' 00134 .ADDRESS P.ABC
00138 P.ABE: .ASCII \RESOURCE_WAIT\<0><0><0>
00147
010E000D 00148 P.ABD: .LONG 17694733
00000000' 0014C .ADDRESS P.ABE
41 57 5F 45 43 52 55 4F 53 45 52 5D 4F 4E 5B 00150 P.ABG: .ASCII \[NO]RESOURCE_WAIT\<0><0><0>
00 00 00 54 49 0015F
010E0011 00164 P.ABF: .LONG 17694737
00000000' 00168 .ADDRESS P.ABG
00 00 53 45 47 45 4C 49 56 49 52 50 0016C P.ABI: .ASCII \PRIVILEGES\<0><0>
010E000A 00178 P.ABH: .LONG 17694730
00000000' 0017C .ADDRESS P.ABI
00 00 53 45 47 45 4C 49 56 49 52 50 00180 P.ABK: .ASCII \PRIVILEGES\<0><0>
010E000A 0018C P.ABJ: .LONG 17694730
00000000' 00190 .ADDRESS P.ABK
00 00 53 45 47 45 4C 49 56 49 52 50 00194 P.ABM: .ASCII \PRIVILEGES\<0><0>
010E000A 001A0 P.ABL: .LONG 17694730
00000000' 001A4 .ADDRESS P.ABM
00 00 53 45 47 45 4C 49 56 49 52 50 001A8 P.ABO: .ASCII \PRIVILEGES\<0><0>
010E000A 001B4 P.ABN: .LONG 17694730
00000000' 001B8 .ADDRESS P.ABO
50 4D 55 44 001BC P.ABQ: .ASCII \DUMP\
010E0004 001C0 P.ABP: .LONG 17694724
00000000' 001C4 .ADDRESS P.ABQ
50 4D 55 44 001C8 P.ABS: .ASCII \DUMP\
010E0004 001CC P.ABR: .LONG 17694724
00000000' 001D0 .ADDRESS P.ABS
```

.PSECT \$CODE\$,NOWRT,2

OFFC 00000 GET_QUALS:

```
5B 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 0326
5A 00000000G 8F D0 00009 MOVAB CLISGET VALUE, R11
59 00000000G 00 9E 00010 MOVL #SET$ OWNPROC, R10
58 00000000G 00 9E 00017 MOVAB LIB$SIGNAL, R9
57 0000' CF 9E 0001E MOVAB CLISPRESENT, R8
5E 34 C2 00023 MOVAB P.AAJ, R7
52 04 AC D0 00026 SUBL2 #52, SP
56 04 A2 9E 0002A MOVL DATA BUFFER, R2 : 0354
55 0C A2 9E 0002E MOVAB 4(R2), R6
54 14 A2 9E 00032 MOVAB 12(R2), R5
53 1C A2 9E 00036 MOVAB 20(R2), R4
50 1C AE 9E 0003A MOVAB 28(R2), R3
80 03190004 8F D0 0003E MOVAB JPI LIST, $$ITMBLKPTR : 0368
80 6E 9E 00045 MOVL #51970052, ($$ITMBLKPTR)+
80 80 7C 00048 MOVAB OURPID, ($$ITMBLKPTR)+
7E 7C 0004A CLRQ ($$ITMBLKPTR)+
34 AE 9F 0004C CLRQ -(SP) : 0370
28 AE 9F 0004F PUSHAB IOSB
7E 7C 00052 PUSHAB JPI LIST
7E D4 00054 CLRQ -(SP)
00000000G 00 07 FB 00056 CALLS #7, SYSSGETJPIW
57 DD 0005D PUSHL R7 : 0376
```

62	01	14	68 01 AE 020E0000 18 14 10	01 50 8F AE AE A7	FB 0005F F0 00062 D0 00067 D4 0006F 9F 00072 9F 00075	CALLS #1, CLISPRES INSV R0, #1, (R2) MOVL #34471936, DESC CLRL DESC+4 PUSHAB DESC PUSHAB P.AAL	:	0381
			68 38 62	02 50 04	FB 00078 E9 0007B 88 0007E	CALLS #2, CLISGET_VALUE BLBC R0, 4\$ BISB2 #4, (R2)	:	0385
			08 1C 1C	A2 AE AE	9F 00081 DD 00084 3C 00087	PUSHAB 8(R2) PUSHL DESC+4 MOVZWL DESC, -(SP)	:	0386 0387 0386
	00000000G		7E 00 05	03 50 A7	FB 0008B E8 00092 9F 00095	CALLS #3, LIB\$CVT_DTB BLBS R0, 1\$ PUSHAB P.AAN	:	0389
			20	0E	11 00098	BRB 3\$:	0392
		1F	08	A2	D1 0009A 1\$:	CMPD 8(R2), #31	:	0393
			08	05	14 0009E	BGTR 2\$:	0394
			30	A2	D5 000A0	TSTL 8(R2)	:	
			18	11	18 000A3	BGEQ 4\$:	
			02	A7	9F 000A5 2\$:	PUSHAB P.AAP	:	
			02	AE	9F 000A8 3\$:	PUSHAB DESC	:	
			02	DD	000AB	PUSHL #2	:	
		0077132A	8F	DD	000AD	PUSHL #7803690	:	
	69		04	FB	000B3	CALLS #4, LIB\$SIGNAL	:	0401
			14	AE	9F 000B6 4\$:	PUSHAB DESC	:	
			3C	A7	9F 000B9	PUSHAB P.AAR	:	
	68		02	FB	000BC	CALLS #2, CLISGET_VALUE	:	
	26		50	E9	000BF	BLBC R0, 6\$:	0404
	66		6E	D1	000C2	CMPD OURPID, (R6)	:	
			0A	13	000C5	BEQL 5\$:	0406
			48	A7	9F 000C7	PUSHAB P.AAT	:	0405
			01	DD	000CA	PUSHL #1	:	
			5A	DD	000CC	PUSHL R10	:	
	69		03	FB	000CE	CALLS #3, LIB\$SIGNAL	:	0408
	62		08	88	000D1 5\$:	BISB2 #8, (R2)	:	0409
	65	14	AE	3C	000D4	MOVZWL DESC, (R5)	:	0410
	04	18	AE	D0	000D8	MOVL DESC+4, 4(R5)	:	0411
	14	AE 020E0000	8F	D0	000DD	MOVL #34471936, DESC	:	
			18	AE	D4 000E5	CLRL DESC+4	:	
			58	A7	9F 000E8 6\$:	PUSHAB P.AAV	:	0418
	68		01	FB	000EB	CALLS #1, CLISPRES	:	
	55		50	D0	000EE	MOVL R0, STATUS	:	0419
	05		55	E9	000F1	BLBC STATUS, 7\$:	0420
	62		20	88	000F4	BISB2 #32, (R2)	:	
			0C	11	000F7	BRB 8\$:	
	00000000G	8F	55	D1	000F9 7\$:	CMPD STATUS, #CLIS_NEGATED	:	0421
			03	12	00100	BNEQ 8\$:	
	62		10	88	00102	BISB2 #16, (R2)	:	0422
		68	A7	9F	00105 8\$:	PUSHAB P.AAX	:	0423
	68		01	FB	00108	CALLS #1, CLISPRES	:	
	03		50	E9	0010B	BLBC R0, 9\$:	
	62		10	88	0010E	BISB2 #16, (R2)	:	0424
		78	A7	9F	00111 9\$:	PUSHAB P.AAZ	:	0429
	68		01	FB	00114	CALLS #1, CLISPRES	:	
	55		50	D0	00117	MOVL R0, STATUS	:	
	00000000G	8F	55	D1	0011A	CMPD STATUS, #CLIS_ABSENT	:	0430

62	01	66	0088	1C 13 00121 6E D1 00123 0B 13 00126 C7 9F 00128 01 DD 0012C 5A DD 0012E 03 FB 00130 8F 88 00133 10\$: 55 D2 00137 50 F0 0013A C7 9F 0013F 11\$: 01 FB 00143 50 D0 00146 55 D1 00149 1C 13 00150 6E D1 00152 0B 13 00155 C7 9F 00157 01 DD 0015B 5A DD 0015D 03 FB 0015F 01 88 00162 12\$: 55 D2 00166 50 F0 00169 C7 9F 0016E 13\$: 01 FB 00172 50 E8 00175 0080 31 00178 6E D1 0017B 14\$: 0B 13 0017E C7 9F 00180 01 DD 00184 5A DD 00186 03 FB 00188 04 88 0018B 15\$: 00 D0 0018F 50 D0 00196 50 D0 0019A 00 D0 0019E 50 D0 001A5 50 D0 001A9 AE 9F 001AD 16\$: C7 9F 001B0 02 FB 001B4 50 E9 001B7 AE 9F 001BA AE 9F 001BD 02 FB 001C0 50 D0 001C7 55 E8 001CA C7 9F 001CD AE 9F 001D1 02 DD 001D4 8F DD 001D6 04 FB 001DC CC 11 001DF AE CB 001E1 17\$:	BEQL 11\$ CMPL OURPID, (R6) BEQL 10\$ PUSHAB P.ABB PUSHL #1 PUSHL R10 CALLS #3, LIB\$SIGNAL BISB2 #64, (R2) MCOML STATUS, R0 INSV R0, #7, #1, (R2) PUSHAB P.ABD CALLS #1, CLIS\$PRESENT MOVL R0, STATUS CMPL STATUS, #CLIS\$_ABSENT BEQL 13\$ CMPL OURPID, (R6) BEQL 12\$ PUSHAB P.ABF PUSHL #1 PUSHL R10 CALLS #3, LIB\$SIGNAL BISB2 #1, 1(R2) MCOML STATUS, R0 INSV R0, #9, #1, (R2) PUSHAB P.ABH CALLS #1, CLIS\$PRESENT BLBS R0, 14\$ BRW 18\$ CMPL OURPID, (R6) BEQL 15\$ PUSHAB P.ABJ PUSHL #1 PUSHL R10 CALLS #3, LIB\$SIGNAL BISB2 #4, 1(R2) MOVL CTL\$GQ PROCPRIV, R0 MOVL R0, NEWPRIV MOVL R0, OLDPRIV MOVL CTL\$GQ PROCPRIV+4, R0 MOVL R0, NEWPRIV+4 MOVL R0, OLDPRIV+4 PUSHAB DESC PUSHAB P.ABL CALLS #2, CLIS\$GET_VALUE BLBC R0, 17\$ PUSHAB NEWPRIV PUSHAB DESC CALLS #2, PRV\$SETPRIV MOVL R0, STATUS BLBS STATUS, 16\$ PUSHAB P.ABN PUSHAB DESC PUSHL #2 PUSHL #7803690 CALLS #4, LIB\$SIGNAL BRB 16\$ BICL3 OLDPRIV, NEWPRIV, (R4)	0433 0435 0434 0437 0438 0444 0445 0448 0450 0449 0452 0453 0459 0465 0467 0466 0469 0474 0475 0484 0487 0489 0488 0484 0496
----	----	----	------	---	--	--

SETPROCES
V04-000

K 4
16-Sep-1984 00:45:54
14-Sep-1984 12:09:16

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETPROCES.B32;1

Page 20
(6)

04	A4	08	AE	10	AE	CB	001E7	BICL3	OLDPRIV+4, NEWPRIV+4, 4(R4)	: 0497
	63	0C	AE	04	AE	CB	001EE	BICL3	NEWPRIV, OLDPRIV, (R3)	: 0498
04	A3	10	AE	08	AE	CB	001F4	BICL3	NEWPRIV+4, OLDPRIV+4, 4(R3)	: 0499
				0118	C7	9F	001FB	18\$: PUSHAB	P.ABP	: 0505
		68			01	FB	001FF	CALLS	#1, CLISPRESNT	
		55			50	D0	00202	MOVL	R0, STATUS	
	00000000G	8F			55	D1	00205	CMPL	STATUS, #CLIS_ABSENT	: 0506
					19	13	0020C	BEQL	20\$	
		66			6E	D1	0020E	CMPL	OURPID, (R6)	: 0509
				0124	0B	13	00211	BEQL	19\$	
					C7	9F	00213	PUSHAB	P.ABR	: 0511
					01	DD	00217	PUSHL	#1	: 0510
					5A	DD	00219	PUSHL	R10	
		69			03	FB	0021B	CALLS	#3, LIB\$SIGNAL	
62		01	A2		08	88	0021E	19\$: BISB2	#8, 1(R2)	: 0513
			0C		55	F0	00222	INSV	STATUS, #12, #1, (R2)	: 0514
					04	00227	20\$: RET			: 0518

; Routine Size: 552 bytes, Routine Base: \$CODE\$ + 0181


```

525 0519 1 ROUTINE set_process (data_buffer) : NOVALUE =
526 0520 2 BEGIN
527 0521 2 ++
528 0522 2
529 0523 2 Set all the parameters specified, signalling any errors.
530 0524 2
531 0525 2 Inputs
532 0526 2     FLAGS will have bits set to indicate what is to change.
533 0527 2     PRIORITY will have the new priority.
534 0528 2     NEW_NAME will point to the new process name.
535 0529 2     PRIV will be the new privilege mask.
536 0530 2
537 0531 2 Outputs
538 0532 2     None
539 0533 2
540 0534 2 --
541 0535 2 MAP
542 0536 2     data_buffer : REF VECTOR;
543 0537 2
544 0538 2 LOCAL
545 0539 2     status;
546 0540 2
547 0541 2
548 0542 2 Bind the data buffer to pleasant, simple names that humans can enjoy
549 0543 2
550 0544 2 bind_data:
551 0545 2
552 0546 2
553 0547 2 /PRIORITY = n
554 0548 2
555 0549 2 IF .flags[set$_priority]
556 0550 2 THEN
557 0551 2     BEGIN
558 0552 2     LOCAL
559 0553 2         want_priority;
560 0554 2
561 P 0555 4     IF NOT (status = $SETPRI(PIDADR = pid,
562 0556 4         PRI = .priority))
563 0557 4     THEN SIGNAL(set$_writeerr, 1, %ASCII 'process priority',
564 0558 4         .status)
565 0559 4
566 0560 3 If the priority requested is greater than the base priority and the process
567 0561 3 does not have ALTPRI privilege then $SETPRI will only set the priority to the
568 0562 3 base. If this is the case or the user requested a log then we need further
569 0563 3 information on the process to tell the user. Since we cannot be sure if the
570 0564 3 wanted priority was set until after the $GETJPIW we must do it in all cases.
571 0565 3
572 0566 3 ELSE
573 0567 4 BEGIN
574 0568 4     LOCAL
575 0569 4         iosb : VECTOR[4,WORD],
576 0570 4         jpi_list : $ITMLST_DECL(ITEMS=2);
577 0571 4
578 0572 4     want_priority = .data_buffer[2];          ! Save the priority requested
579 0573 4                                           ! Generic value because of BIND
580 0574 4
581 0575 4 ! Set up the JPI item list to get the new process priority.

```

```
582      0576 4 !
583      P 0577 4 $ITMLST_INIT(ITMLST = jpi_list,
584      0578 4 (ITMCOD = jpi$_prio, BUFADR = priority));
585      P 0579 4 status = $GETJPIW(ITMLST = jpi_list,
586      P 0580 4 PIDADR = pid,
587      0581 4 IOSB = iosb);
588      0582 4 IF .status
589      0583 4 THEN status = .iosb[0];
590      0584 4 IF NOT .status
591      0585 4 THEN SIGNAL(.status)
592      0586 4 ELSE
593      0587 4 !
594      0588 4 Display correct message
595      0589 4 !
596      0590 5 BEGIN
597      0591 5 IF .want_priority GTR .priority ! If the desired priority was not
598      0592 5 THEN SIGNAL(set$_noprio) ! set then ALTPRI not set
599      0593 5 ELSE
600      0594 5 IF .flags[set$_log] ! If logging requested
601      0595 5 THEN SIGNAL(set$_prio, 3, name_desc, .pid, .priority);
602      0596 4 END;
603      0597 3 END;
604      0598 2 END;
605      0599 2 !
606      0600 2 !
607      0601 2 /NAME = string
608      0602 2 !
609      0603 2 IF .flags[set$_name]
610      0604 2 THEN
611      0605 3 BEGIN
612      0606 4 IF NOT (status = $SETPRN(PCNAM = new_name))
613      0607 4 THEN SIGNAL(set$_writeerr, 1, ! Signal if an error
614      0608 4 %ASCII 'process name',
615      0609 4 .status)
616      0610 4 ELSE IF .flags[set$_log]
617      0611 4 THEN SIGNAL(set$_nameset, 1, new_name); ! or if /LOG
618      0612 4 END;
619      0613 3 !
620      0614 2 !
621      0615 2 /SUSPEND
622      0616 2 !
623      0617 2 IF .flags[set$_suspend]
624      0618 2 THEN
625      0619 3 BEGIN
626      0620 4 IF NOT (status = $SUSPND(PIDADR = pid)) ! If a problem,
627      0621 4 THEN SIGNAL(set$_notsuspend, 2, name_desc, .pid, ! signal it
628      0622 4 .status)
629      0623 4 ELSE IF .flags[set$_log] ! If /LOG, signal it
630      0624 4 THEN SIGNAL(set$_suspend, 2, name_desc, .pid);
631      0625 4 END;
632      0626 3 !
633      0627 2 !
634      0628 2 /NOSUSPEND or /RESUME
635      0629 2 !
636      0630 2 IF .flags[set$_resume]
637      0631 2 THEN
638      0632 3 BEGIN
```



```

639 0633 4 IF NOT (status = $RESUME(PIDADR = pid))
640 0634 3 THEN SIGNAL(set$_notresumed, 2, name_desc, .pid, .status)
641 0635 3 ELSE IF .flags[set$_log]
642 0636 3 THEN SIGNAL(set$_resumed, 2, name_desc, .pid);
643 0637 3 END;
644 0638 3
645 0639 3
646 0640 3 /[/NO]SWAP
647 0641 3
648 0642 3 IF .flags[set$_swap]
649 0643 3 THEN
650 0644 3 BEGIN
651 0645 4 IF NOT (status = $SETSWM(SWPFLG = .flags[set$_swapval])) ! If an error,
652 0646 4 THEN SIGNAL(set$_writeerr, 1, %ASCII 'swap mode', ! signal it
653 0647 4 .status)
654 0648 3 ELSE IF .flags[set$_log]
655 0649 3 THEN SIGNAL(set$_modeset, 1,
656 0650 4 (IF .flags[set$_swapval]
657 0651 4 THEN %ASCII 'NOSWAP'
658 0652 4 ELSE %ASCII 'SWAP')));
659 0653 3 END;
660 0654 3
661 0655 3 /[/NO]RESOURCE_WAIT
662 0656 3
663 0657 3 IF .flags[set$_wait]
664 0658 3 THEN
665 0659 3 BEGIN
666 0660 4 IF NOT (status = $SETRWM(WATFLG = .flags[set$_waitval]))
667 0661 4 THEN SIGNAL(set$_writeerr, 1, ! Signal if a problem
668 0662 4 %ASCII 'resource wait mode',
669 0663 4 .status)
670 0664 3 ELSE IF .flags[set$_log] ! Signal if /LOG
671 0665 3 THEN SIGNAL(set$_modeset, 1,
672 0666 4 (IF .flags[set$_waitval]
673 0667 4 THEN %ASCII 'NORESOURCE_WAIT'
674 0668 4 ELSE %ASCII 'RESOURCE_WAIT'));
675 0669 3 END;
676 0670 3
677 0671 3
678 0672 3 /[/PRIVILEGES = list
679 0673 3
680 0674 3 IF .flags[set$_priv]
681 0675 3 THEN
682 0676 3 BEGIN
683 0677 3
684 0678 3
685 0679 3 Enable the new privileges.
686 0680 3
687 0681 3
688 0682 3 IF .enab_priv[0] NEQ 0 ! If anything to enable,
689 0683 3 OR .enab_priv[1] NEQ 0 ! do it and save the status,
690 0684 3 THEN status = $SETPRV(PRVADR = enab_priv, !
691 0685 3 PRMFLG = 1,
692 0686 3 ENBFLG = 1)
693 0687 3 ELSE status = 1; ! otherwise set success.
694 0688 3
695 0689 3 IF .disab_priv[0] NEQ 0 ! If anything to disable,
```



```

696      OR .disab_priv[1] NEQ 0
697      P 0691 THEN $SETPRV(PRVADR = disab_priv,
698      P 0692     PRMFLG = 1,
699      0693     ENBFLG = 0);
700      0694
701      0695 IF NOT .status
702      0696 THEN SIGNAL(set$_notpriv, .status)
703      0697 ELSE
704      0698 BEGIN
705      0699     IF .status EQL ss$_notallpriv
706      0700     THEN SIGNAL(ss$_notallpriv AND %X'FFFFFFFE')
707      0701     ELSE IF .flags[set$_log]
708      0702     THEN SIGNAL(set$_privset);
709      0703     END;
710      0704 END;
711      0705
712      0706 /[NO]DUMP
713      0707
714      0708 IF .flags[set$_dump]
715      0709 THEN
716      0710     IF .flags[set$_dumpval]
717      0711     THEN
718      0712 BEGIN
719      0713     $CMKRNLC(ROUTIN = set_dump,
720      P 0714     ARGST = UPLIT(1,1));
721      0715     SIGNAL(set$_modeset, 1, %ASCII 'DUMP');
722      0716     END
723      0717 ELSE
724      0718 BEGIN
725      0719     $CMKRNLC(ROUTIN = set_dump,
726      P 0720     ARGST = UPLIT(1,0));
727      0721     SIGNAL(set$_modeset, 1, %ASCII 'NODUMP');
728      0722     END;
729      0723
730      0724 RETURN;
731      0725
732      0726 1 END;
```

```

.PSECT $PLITS,NOWRT,NOEXE,2
74 69 72 6F 69 72 70 20 73 73 65 63 6F 72 70 001D4 P.ABU: .ASCII \process priority\
79 001E3
010E0010 001E4 P.ABT: .LONG 17694736
00000000 001E8 .ADDRESS P.ABU
65 6D 61 6E 20 73 73 65 63 6F 72 70 001EC P.ABW: .ASCII \process name\
010E000C 001F8 P.ABV: .LONG 17694732
00000000 001FC .ADDRESS P.ABW
00 00 00 65 64 6F 6D 20 70 61 77 73 00200 P.ABY: .ASCII \swap mode\<0><0><0>
010E0009 0020C P.ABX: .LONG 17694729
00000000 00210 .ADDRESS P.ABY
00 00 50 41 57 53 4F 4E 00214 P.ACA: .ASCII \NOSWAP\<0><0>
010E0006 0021C P.ABZ: .LONG 17694726
00000000 00220 .ADDRESS P.ACA
50 41 57 53 00224 P.ACC: .ASCII \SWAP\
010E0004 00228 P.ACB: .LONG 17694724
```



```
6D 20 74 69 61 77 20 65 63 72 75 6F 73 65 72 0022C .ADDRESS P.ACC
00 00 65 64 6F 00230 P.ACE: .ASCII \resource wait mode\<0><0>
010E0012 0023F
00000000 00244 P.ACD: .LONG 17694738
00000000 00248 .ADDRESS P.ACE
54 49 41 57 5F 45 43 52 55 4F 53 45 52 4F 4E 0024C P.ACG: .ASCII \NORESOURCE_WAIT\<0>
00 0025B
010E000F 0025C P.ACF: .LONG 17694735
00000000 00260 .ADDRESS P.ACG
00 00 54 49 41 57 5F 45 43 52 55 4F 53 45 52 00264 P.ACI: .ASCII \RESOURCE_WAIT\<0><0><0>
00 00273
010E000D 00274 P.ACH: .LONG 17694733
00000000 00278 .ADDRESS P.ACI
00000001 0027C P.ACJ: .LONG 1, 1
50 4D 55 44 00284 P.ACL: .ASCII \DUMP\
010E0004 00288 P.ACK: .LONG 17694724
00000000 0028C .ADDRESS P.ACL
00000000 00290 P.ACM: .LONG 1, 0
00 00 50 4D 55 44 4F 4E 00298 P.ACO: .ASCII \NODUMP\<0><0>
010E0006 002A0 P.ACN: .LONG 17694726
00000000 002A4 .ADDRESS P.ACO
```

```
.EXTRN SYSS$SETPRI, SYSS$SETPRN
.EXTRN SYSS$SUSPND, SYSS$RESUME
.EXTRN SYSS$SETSWM, SYSS$SETRWM
.EXTRN SYSS$SETPRV, SYSS$CMKRNL
```

```
.PSECT $CODE$,NOWRT,2
```

OFFC 00000 SET_PROCESS:

```
5B 00000000G 8F D0 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
5A 0000 0000 CF 9E 00009 MOVL #SET$WRITEERR, R11
59 00000000G 00 9E 0000E MOVAB P.ABT, R10
5E 24 C2 00015 MOVAB LIB$SIGNAL, R9
52 04 AC D0 00018 SUBL2 #36, SP
53 04 A2 9E 0001C MOVL DATA_BUFFER, R2
57 14 A2 9E 00020 MOVAB 4(R2), R3
56 1C A2 9E 00024 MOVAB 20(R2), R7
55 24 A2 9E 00028 MOVAB 28(R2), R6
6C 62 02 E1 0002C MOVAB 36(R2), R5
7E D4 00030 BBC #2, (R2), 5$
08 A2 DD 00032 CLRL -(SP)
7E D4 00035 PUSH 8(R2)
53 DD 00037 CLRL -(SP)
00 04 FB 00039 PUSH R3
54 50 D0 00040 CALLS #4, SYSS$SETPRI
0D 54 E8 00043 MOV R0, STATUS
54 DD 00046 BLBS STATUS, 1$
5A DD 00048 PUSH STATUS
01 DD 0004A PUSH R10
5B DD 0004C PUSH #1
69 04 FB 0004E PUSH R11
58 08 A2 D0 00053 CALLS #4, LIB$SIGNAL
50 6E 9E 00057 BRB 7$
80 03090004 8F D0 0005A MOV 8(R2), WANT_PRIORITY
1$: MOVAB JPI LIST, $SITMBLKPTR
MOVL #50921476, ($SITMBLKPTR)+
```


	80	08	A2	9E	00061	MOVAB	8(R2), (\$\$ITMBLKPTR)+		
			80	7C	00065	CLRQ	(\$\$ITMBLKPTR)+		
			7E	7C	00067	CLRQ	-(SP)		0581
		24	AE	9F	00069	PUSHAB	IOSB		
		0C	AE	9F	0006C	PUSHAB	JPI LIST		
			7E	D4	0006F	CLRL	-(SP)		
			53	DD	00071	PUSHL	R3		
			7E	D4	00073	CLRL	-(SP)		
00000000G	00		07	FB	00075	CALLS	#7, SYSSGETJPIW		
	54		50	DD	0007C	MOVL	R0, STATUS		
	07		54	E9	0007F	BLBC	STATUS, 2\$		0582
	54	1C	AE	3C	00082	MOVZWL	IOSB, STATUS		0583
	04		54	E8	00086	BLBS	STATUS, 3\$		0584
			54	DD	00089	PUSHL	STATUS		0585
			0C	11	0008B	BRB	4\$		
08	A2		58	D1	0008D	3\$:	CMP	WANT_PRIORITY, 8(R2)	0591
			0B	15	00091	BLEQ	6\$		
	00000000G		8F	DD	00093	PUSHL	#SETS NOPRIO		0592
	69		01	FB	00099	4\$:	CALLS	#1, LIBSSIGNAL	
			16	11	0009C	5\$:	BRB	7\$	
12	62		01	E1	0009E	6\$:	BBC	#1, (R2), 7\$	0594
		08	A2	DD	000A2		PUSHL	8(R2)	0595
			63	DD	000A5		PUSHL	(R3)	
			55	DD	000A7		PUSHL	R5	
			03	DD	000A9		PUSHL	#3	
	00000000G		8F	DD	000AB		PUSHL	#SETS PRIOSET	
	69		05	FB	000B1		CALLS	#5, LIBSSIGNAL	
30	62		03	E1	000B4	7\$:	BBC	#3, (R2), 9\$	0603
		0C	A2	9F	000B8		PUSHAB	12(R2)	0606
	00000000G		01	FB	000BB		CALLS	#1, SYSSSETPRN	
			50	DD	000C2		MOVL	R0, STATUS	
	54		54	E8	000C5		BLBS	STATUS, 8\$	
	0E		54	DD	000C8		PUSHL	STATUS	0609
		14	AA	9F	000CA		PUSHAB	P.ABV	0607
			01	DD	000CD		PUSHL	#1	
			5B	DD	000CF		PUSHL	R11	
	69		04	FB	000D1		CALLS	#4, LIBSSIGNAL	
			12	11	000D4		BRB	9\$	
0E	62		01	E1	000D6	8\$:	BBC	#1, (R2), 9\$	0610
		0C	A2	9F	000DA		PUSHAB	12(R2)	0611
			01	DD	000DD		PUSHL	#1	
	00000000G		8F	DD	000DF		PUSHL	#SETS NAMESET	
	69		03	FB	000E5		CALLS	#3, LIBSSIGNAL	
37	62		05	E1	000E8	9\$:	BBC	#5, (R2), 11\$	0617
			7E	D4	000EC		CLRL	-(SP)	0620
			53	DD	000EE		PUSHL	R3	
	00000000G		02	FB	000F0		CALLS	#2, SYSSSUSPND	
			50	DD	000F7		MOVL	R0, STATUS	
	54		54	E8	000FA		BLBS	STATUS, 10\$	
	13		54	DD	000FD		PUSHL	STATUS	0622
			63	DD	000FF		PUSHL	(R3)	0621
			55	DD	00101		PUSHL	R5	
			02	DD	00103		PUSHL	#2	
	00000000G		8F	DD	00105		PUSHL	#SETS NOTSUSPND	
	69		05	FB	0010B		CALLS	#5, LIBSSIGNAL	
			13	11	0010E		BRB	11\$	
0F	62		01	E1	00110	10\$:	BBC	#1, (R2), 11\$	0623

		63	DD	00114	PUSHL	(R3)	0624
		55	DD	00116	PUSHL	R5	
		02	DD	00118	PUSHL	#2	
	00000000G	8F	DD	0011A	PUSHL	#SETS, SUSPND	
37	69	04	FB	00120	CALLS	#4, LIB\$SIGNAL	
	62	04	E1	00123	BBC	#4, (R2), 13\$	0630
		7E	D4	00127	CLRL	-(SP)	0633
		53	DD	00129	PUSHL	R3	
	00000000G	02	FB	0012B	CALLS	#2, SYS\$RESUME	
	54	50	DO	00132	MOVL	R0, STATUS	
	13	54	E8	00135	BLBS	STATUS, 12\$	
		54	DD	00138	PUSHL	STATUS	0634
		63	DD	0013A	PUSHL	(R3)	
		55	DD	0013C	PUSHL	R5	
		02	DD	0013E	PUSHL	#2	
	00000000G	8F	DD	00140	PUSHL	#SETS, NOTRESUMED	
	69	05	FB	00146	CALLS	#5, LIB\$SIGNAL	
		13	11	00149	BRB	13\$	
0F	62	01	E1	0014B	BBC	#1, (R2), 13\$	0635
		63	DD	0014F	PUSHL	(R3)	0636
		55	DD	00151	PUSHL	R5	
		02	DD	00153	PUSHL	#2	
	00000000G	8F	DD	00155	PUSHL	#SETS, RESUMED	
	69	04	FB	0015B	CALLS	#4, LIB\$SIGNAL	
7E	3F	06	E1	0015E	BBC	#6, (R2), 17\$	0642
	62	07	EF	00162	EXTZV	#7, #1, (R2), -(SP)	0645
		01	FB	00167	CALLS	#1, SYS\$SETSWM	
	00000000G	50	DO	0016E	MOVL	R0, STATUS	
	54	54	E8	00171	BLBS	STATUS, 14\$	
	0E	54	DD	00174	PUSHL	STATUS	0647
		28	AA	9F	PUSHAB	P.ABX	0646
		01	DD	00179	PUSHL	#1	
		5B	DD	0017B	PUSHL	R11	
	69	04	FB	0017D	CALLS	#4, LIB\$SIGNAL	
		1F	11	00180	BRB	17\$	
1B	62	01	E1	00182	BBC	#1, (R2), 17\$	0648
		62	95	00186	TSTB	(R2)	0650
		06	18	00188	BGEQ	15\$	
	50	38	AA	9E	MOVAB	P.ABZ, R0	0651
		04	11	0018E	BRB	16\$	
	50	44	AA	9E	MOVAB	P.ACB, R0	0652
		50	DD	00194	PUSHL	R0	
		01	DD	00196	PUSHL	#1	0649
	00000000G	8F	DD	00198	PUSHL	#SETS, MODESET	
	69	03	FB	0019E	CALLS	#3, LIB\$SIGNAL	
7E	40	01	A2	E9	BLBC	1(R2), 21\$	0658
	01	09	EF	001A5	EXTZV	#9, #1, (R2), -(SP)	0661
		01	FB	001AA	CALLS	#1, SYS\$SETRWM	
	00000000G	50	DO	001B1	MOVL	R0, STATUS	
	54	54	E8	001B4	BLBS	STATUS, 18\$	
	0E	54	DD	001B7	PUSHL	STATUS	0664
		60	AA	9F	PUSHAB	P.ACD	0662
		01	DD	001BC	PUSHL	#1	
		5B	DD	001BE	PUSHL	R11	
	69	04	FB	001C0	CALLS	#4, LIB\$SIGNAL	
		20	11	001C3	BRB	21\$	
1C	62	01	E1	001C5	BBC	#1, (R2), 21\$	0665

06	62	09	E1	001C9	BBC	#9, (R2), 19\$	0667
	50	AA	9E	001CD	MOVAB	P.ACF, R0	0668
		05	11	001D1	BRB	20\$	
	50	CA	9E	001D3	MOVAB	P.ACH, R0	0669
		50	DD	001D8	PUSHL	R0	
		01	DD	001DA	PUSHL	#1	0666
	00000000G	8F	DD	001DC	PUSHL	#SET\$ MODESET	
	69	03	FB	001E2	CALLS	#3, LIB\$SIGNAL	
63	62	0A	E1	001E5	BBC	#10, (R2), 30\$	0675
		67	D5	001E9	TSTL	(R7)	0682
		05	12	001EB	BNEQ	22\$	
		A7	D5	001ED	TSTL	4(R7)	0683
		13	13	001F0	BEQL	23\$	
	7E	01	7D	001F2	MOVQ	#1, -(SP)	0686
		57	DD	001F5	PUSHL	R7	
		01	DD	001F7	PUSHL	#1	
00000000G	00	04	FB	001F9	CALLS	#4, SYS\$SETPRV	
	54	50	DD	00200	MOVL	R0, STATUS	
		03	11	00203	BRB	24\$	0684
	54	01	DD	00205	MOVL	#1, STATUS	0687
		66	D5	00208	TSTL	(R6)	0689
		05	12	0020A	BNEQ	25\$	
		A6	D5	0020C	TSTL	4(R6)	0690
		0E	13	0020F	BEQL	26\$	
	7E	01	7D	00211	MOVQ	#1, -(SP)	0693
		56	DD	00214	PUSHL	R6	
		7E	D4	00216	CLRL	-(SP)	
00000000G	00	04	FB	00218	CALLS	#4, SYS\$SETPRV	
	0D	54	E8	0021F	BLBS	STATUS, 27\$	0695
		54	DD	00222	PUSHL	STATUS	0696
	00000000G	8F	DD	00224	PUSHL	#SET\$ NOTPRIV	
	69	02	FB	0022A	CALLS	#2, LIB\$SIGNAL	
		1D	11	0022D	BRB	30\$	
00000681	8F	54	D1	0022F	CMPL	STATUS, #1665	0699
		07	12	00236	BNEQ	28\$	
	7E	8F	3C	00238	MOVZWL	#1664, -(SP)	0700
		0A	11	0023D	BRB	29\$	
09	62	01	E1	0023F	BBC	#1, (R2), 30\$	0701
		8F	DD	00243	PUSHL	#SET\$ PRIVSET	0702
	69	01	FB	00249	CALLS	#1, LIB\$SIGNAL	
37	62	0B	E1	0024C	BBC	#11, (R2), 33\$	0709
15	62	0C	E1	00250	BBC	#12, (R2), 31\$	0711
		CA	9F	00254	PUSHAB	P.ACJ	0715
	0098	CF	9F	00258	PUSHAB	SET_DUMP	
00000000G	00	02	FB	0025C	CALLS	#2, -SYS\$CMKRNL	
		CA	9F	00263	PUSHAB	P.ACK	0716
		13	11	00267	BRB	32\$	
		CA	9F	00269	PUSHAB	P.ACM	0721
	00AC	CF	9F	0026D	PUSHAB	SET_DUMP	
00000000G	00	02	FB	00271	CALLS	#2, -SYS\$CMKRNL	
		CA	9F	00278	PUSHAB	P.ACN	0722
		01	DD	0027C	PUSHL	#1	
	00000000G	8F	DD	0027E	PUSHL	#SET\$ MODESET	
	69	03	FB	00284	CALLS	#3, LIB\$SIGNAL	
		04		00287	RET		0726

; Routine Size: 648 bytes, Routine Base: \$CODE\$ + 03A9

SETPROCES
V04-000

G 5
16-Sep-1984 00:45:54
14-Sep-1984 12:09:16

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETPROCES.B32;1

Page 29
(7)

```

: 734      0727 1 ROUTINE set_dump (mode) : NOVALUE =
: 735      0728 2 BEGIN
: 736      0729 2 ++
: 737      0730 2 Functional description
: 738      0731 2
: 739      0732 2 This routine sets the dump mode. It can only affect the current
: 740      0733 2 process.
: 741      0734 2
: 742      0735 2 Inputs
: 743      0736 2 mode - 1 or 0 for mode on or off
: 744      0737 2
: 745      0738 2 Outputs
: 746      0739 2 None
: 747      0740 2
: 748      0741 2 ----
: 749      0742 2
: 750      0743 2 ctl$gl_phd[phd$v_imgdmp] = .mode;
: 751      0744 2
: 752      0745 2 return;
: 753      0746 2
: 754      0747 1 END;

```

0000 00000 SET_DUMP:

36	A0	01	50	00000000G	00	D0	00002	.WORD	Save nothing	
			05	04	AC	F0	00009	MOVL	CTL\$GL_PHD, R0	: 0727
						04	00010	INSV	MODE, #5, #1, 54(R0)	: 0743
								RET		: 0747

; Routine Size: 17 bytes, Routine Base: \$CODE\$ + 0631

; 755 0748 1

SETPROCES
V04-000

1 5
16-Sep-1984 00:45:54
14-Sep-1984 12:09:16

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETPROCES.B32;1

Page 31
(9)

: 757 0749 1 END
: 758 0750 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$PLITS	680	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODE\$	1602	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	43	0	1000	00:01.8

: Information: 1
: Warnings: 0
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:SETPROCES/OBJ=OBJ\$:SETPROCES MSRC\$:SETPROCES/UPDATE=(ENH\$:SETPROCES)

: Size: 1602 code + 680 data bytes
: Run Time: 00:31.1
: Elapsed Time: 01:46.8
: Lines/CPU Min: 1447
: Lexemes/CPU-Min: 22162
: Memory Used: 228 pages
: Compilation Complete

0054 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

